Chapter 5
On Integrating EAST-ADL and UPPAAL for Embedded System Architecture Verification

Tahir Naseer Qureshi, De-Jiu Chen, Magnus Persson and Martin Törngren

Abstract Model-based development (MBD) is a common approach adopted in many engineering disciplines for handling complexity. For distributed microprocessor based systems MBD approaches include the use of architecture description languages (ADL's), modeling and simulation tools and tools for formal verification. To increase their combined effectiveness, the various MBD methods, tools and languages are required to be integrated with each other. This chapter addresses the connection between ADL's and formal verification in the context of automotive embedded systems. A template-based mapping scheme providing formal interpretation of EAST-ADL, an automotive specific ADL with timed automata (TA) is the main contribution providing a possibility of automated analysis of timing constraints specified for the execution behavior and events of a system. One benefit of using TA is the fact that it can also be used for generating test cases for their usage during late development phases.

5.1 Introduction

The complexity related to automotive embedded systems has increased significantly during the last few decades. While a product is composed of a large number of interconnected software and hardware components, the development process

T. N. Qureshi (✉) · D. J. Chen · M. Persson · M. Törngren
Department of Machine Design, KTH - The Royal Institute of Technology, Stockholm, Sweden
e-mail: tnqu@md.kth.se

D.-J. Chen
e-mail: chen@md.kth.se

M. Persson
e-mail: magnper@md.kth.se

M. Törngren
e-mail: martin@md.kth.se

is also distributed spatially and temporally among different engineering teams. A model-based development (MBD) approach, i.e., the use of computerized models for different activities [1] is followed in various engineering domains to manage complexity. For embedded systems several powerful but disconnected MBD solutions like architectural description languages (ADL) for managing engineering information, tools like Matlab/Simulink and formal methods like model checking for various kind of analysis are used. Furthermore, a lot of faults are discovered during the late development stages (i.e., integration and testing) resulting in an increased development time and cost [2]. Inconsistency between the behavioral specifications, related constraints and requirements is one of the major factors contributing to the faults. A seamless integrated development environment incorporating several model-based methods and tools is envisioned to deal with the issues [3].

This chapter is motivated by the above mentioned needs for a seamless integrated development environment. The main objective is an architecture-centric analysis support for architectural specifications in the context of model-based development of automotive embedded systems. A method which paves a way for automated model transformation between Electronics Architecture and Software Technology- ADL (EAST-ADL); an automotive specific ADL [4] and timed automata (a formalism for verifying real-time systems) [5] is introduced. EAST-ADL provides a multi-viewed and structured information management support at multiple abstraction levels. As compared to languages like UML or AADL, EAST-ADL has a broader coverage in terms of development life-cycle. The support is both product-related, such as hardware, software and infrastructure, as well as concern related, such as requirements, safety, dependability etc. Timed automata (TA) on the other hand is a widely used formalisms for verifying real-time systems.

Provision of a generalized solution for analyzing specifications based on EAST-ADL's design level of abstraction is the main focus of the chapter. Furthermore, the work focuses on domain specific language and generic automotive systems which is in contrast with other approaches like [6, 7] focusing on either a generic language like UML or requirements organized in the form of AND/OR hierarchies.

The chapter mainly presents a mapping framework based on predefined timed automata templates. It is shown that EAST-ADL timing constraint specifications and the execution behavior of a component can be abstracted as a network of timed automata. The considerations required for the usage of the mapping scheme are also discussed. A case study of a brake-by-wire system is used to demonstrate the usage of the framework with PapyrusUML (a UML modeling tool) and UPPAAL (a TA-based model checker) as the tools for modeling and analysis respectively.

5.2 Related Work

Verification based on timed automata (TA) of automotive applications is presented in [8]. The TA models were used to derive the templates related to function execution in this chapter.

A TA-based analysis method for analyzing embedded system architectures is presented in [6]. TA models are derived from UML sequence diagrams augmented with performance data. The TA models are application specific, limited to only a few message lengths for the communication bus, etc. In addition, the generalization of the work is yet to be determined especially with the presence of *variation points* in sequence diagrams. In contrast, the work in this chapter can be applied to multiple applications focusing on EAST-ADL which does not not have variation points like UML.

A number of efforts have been carried out to enable the analysis, verification and validation of system architecture design captured in EAST-ADL. This work is an extension of [9] and [10]. In [9] the work was limited to the reaction time constraint and evaluation of the possibility of model transformation between EAST-ADL and UPPAAL. This chapter presents a few selected templates from [10] and provides an account of experiences and observations.

In [11] an effort to integrate the SPIN model checker for formal verification of EAST-ADL models is presented. The automata addressed by SPIN are untimed and therefore, less suitable for the execution behavior targeted in this chapter.

The specification and analysis of EAST-ADL models can be carried out in several ways. One way is to specify the structure using its core part and define behavior related aspects using external representations such as Simulink. This approach is used by Wang et al. [12] and Enoiu et al. [13] where timed automata is used as the external representation of EAST-ADL functional behavior. An issue with this approach is the gap between different behavioral models used during different development stages by different experts. For example, relation between timed automata models for formal verification and Simulink models for analysis and code generation. A more suitable approach is the usage of EAST-ADL native specifications for defining a common behavior and interpreting them with different formalisms used for different purposes. This approach enables the utilization of the benefits provided by EAST-ADL to a greater extent. This chapter follows the latter approach with focus on the timing model part of EAST-ADL. In [14] we have also complemented the work by addressing the internal behavior of EAST-ADL functions by transforming the EAST-ADL behavioral specifications to timed automata.

5.3 EAST-ADL and Timing Extension - Concept and Notations

EAST-ADL [4] evolved through several European projects since 2001. It complements the best industrial practices of requirements specification, system design, documentation etc. with the goal to facilitate the management of engineering information throughout a vehicle's development life cycle. The language is modular with a core part for specifying structure, and extensions for the specification of logical and execution behavior, requirements, product variations etc. This modular approach not only separates the definition of functional and non-functional aspects but also enables the use of existing tools for various development activities.

A system can be specified atfour different abstraction levels (namely vehicle, analysis, design and implementation) using EAST-ADL. For example, product line features (end-to-end functionality) and their variations are specified at the vehicle level, whereas the detailed design of functional components, connections and allocations to various hardware components is carried out at the design level. The subset of EAST-ADL addressed in this chapter is as follows.

5.3.1 EAST-ADL Core and Behavior Model

The core and behavior artifacts focused in this chapter are shown in Fig. 5.1 where the artifacts prefixed with *Behavior*:: are part of the behavior model. A concept of type and prototype is used for reusability. For example, a function representing a wheel can be instantiated as a prototype within another function representing the overall vehicle. The behavior model is mainly used to specify operational modes and triggering policy for a function or its prototype. For logical behavior (i.e., internal logics or a transfer-function between the inputs and outputs) of a function EAST-ADL relies on external representations like Simulink and SCADE.

5.3.2 Function Behavior Semantics

Every time a function is triggered it reads data at its *input* ports, *executes* computations and writes data on its *output* ports. The EAST-ADL specifications specify concurrent

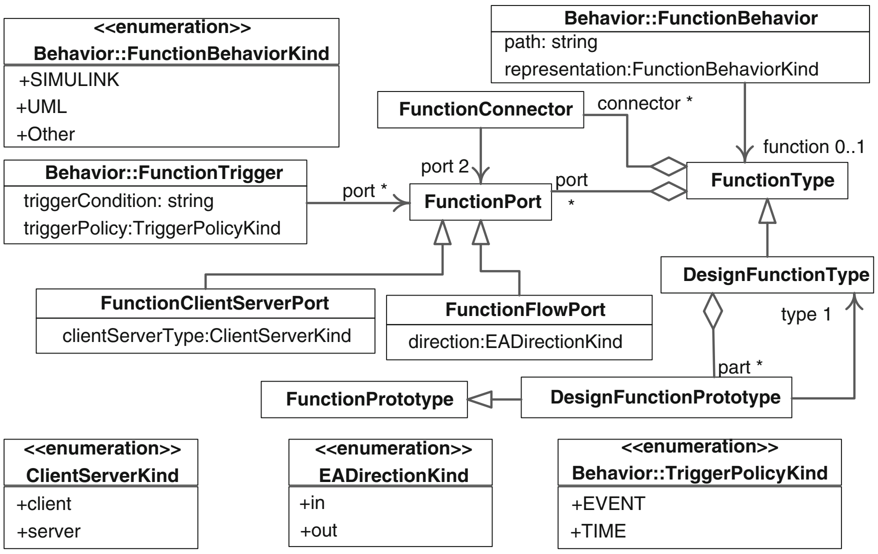

Fig. 5.1 EAST-ADL core structure and behavior model

execution and run-to-completion semantics of a function. i.e., a function runs all the steps before it starts to execute again. The behavior in terms of interruptions and preemption is considered as a part of implementation level of abstraction which is out of scope of the EAST-ADL specifications. Moreover, the ports of an EAST-ADL function have single-sized overwritable and non-consumable buffer semantics.

5.3.3 Timing Model

The timing model of EAST-ADL is derived from Time Augmented Description Language (TADL). The readers are referred to [15] for a conceptual overview of TADL. As shown in Fig. 5.2, the timing extension is based on the concepts of *event* and *event chain*. *EventFunction*, *EventFunctionFlowPort* and *EventFunction-ClientServerPort* are the three event kinds, referring to the triggering of a function by some sort of dispatcher, arrival of a data and service requested (or received) by a port. The dashed *instanceRef* relations indicate the context dependency of an artifact. For example, an EventFunctionClientServerPort is applicable on ports of prototypes instead of their respective function types. An event chain comprises of one or more *stimulus* and *response* events. An event chain can further be refined into smaller event chains called *strands* (parallel chains) or *segments* (sequenced).

Timing constraints can be specified on function executions (e.g. execution time, period, execution time budgets), event occurrences (e.g. arrival or departure of data on a function port) and event chains. The constraints addressed in this chapter are shown in Fig. 5.3. *PeriodicEventConstraint* and *DelayConstraint* in Fig. 5.3 are examples of the constraints applied on event and event chain respectively. For additional information, the readers are referred to [4].

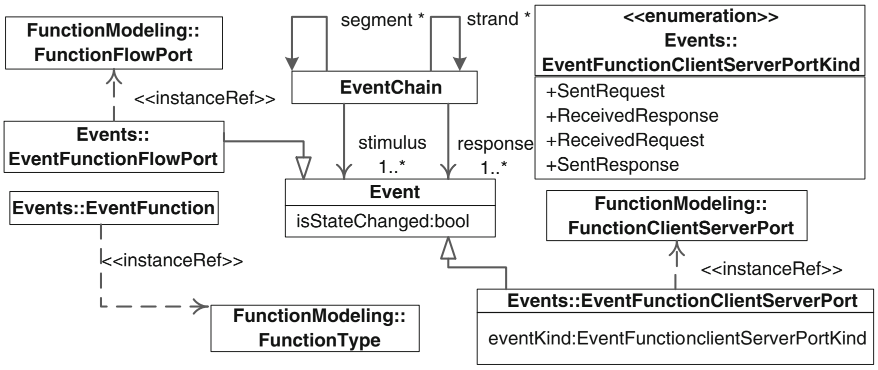

Fig. 5.2 Events and event chains in EAST-ADL

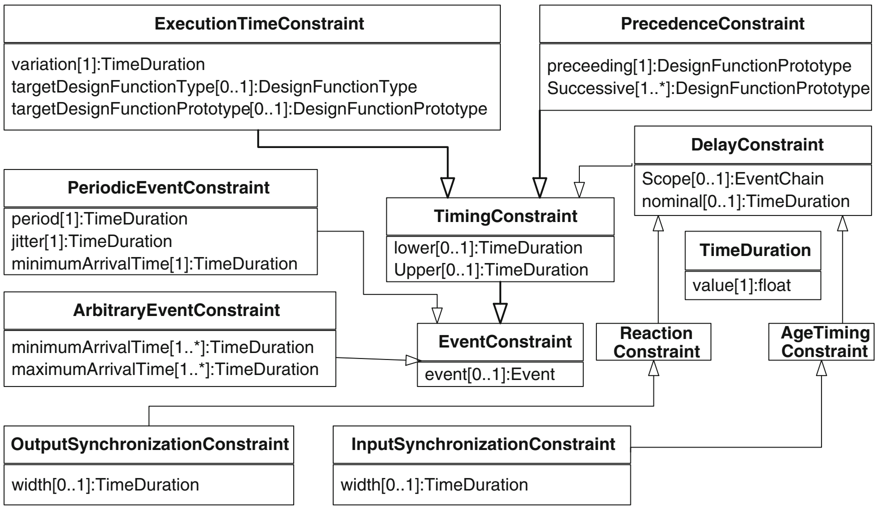
ExecutionTimeConstraint
variation[1]:TimeDuration
targetDesignFunctionType[0..1]:DesignFunctionType
targetDesignFunctionPrototype[0..1]:DesignFunctionPrototype
PrecedenceConstraint
preceeding[1]:DesignFunctionPrototype
Successive[1..*]:DesignFunctionPrototype
DelayConstraint
Scope[0..1]:EventChain
nominal[0..1]:TimeDuration
PeriodicEventConstraint
period[1]:TimeDuration
jitter[1]:TimeDuration
minimumArrivalTime[1]:TimeDuration
TimingConstraint
lower[0..1]:TimeDuration
Upper[0..1]:TimeDuration
TimeDuration
value[1]:float
ArbitraryEventConstraint
minimumArrivalTime[1..*]:TimeDuration
maximumArrivalTime[1..*]:TimeDuration
EventConstraint
event[0..1]:Event
Reaction Constraint
AgeTiming Constraint
OutputSynchronizationConstraint
width[0..1]:TimeDuration
InputSynchronizationConstraint
width[0..1]:TimeDuration

Fig. 5.3 EAST-ADL timing constraints

5.4 Timed Automata and UPPAAL

A timed automaton (TA) [5] is an automaton augmented with clocks and time semantics to enable formal analysis of real-time systems. For example, to specify the maximum time duration for which a location (or state) can remain active, a clock invariant is used. Similarly, it is also possible to specify guards based on clock values on transitions between two locations.

Often a set of TA are used in a networked form with a common set of clocks and actions. A special synchronization action denoted by an exclamation sign (!) or a question mark (?) is used for synchronization between different TA. A timed-automaton in a network is concurrent unless and until mechanisms like synchronization actions are applied. The readers are referred to [5] for a formal definition and semantics of a network of TA.

UPPAAL is a model checker based on TA for modeling, validation and verification of real-time systems. The tool has three main parts: an editor, a simulator and a verifier, for modeling, debugging and verification (covering exhaustive dynamic behavior) respectively. A system in UPPAAL is modeled as a network of TA. A subset of CTL (computation tree logic) is used as the query language in UPPAAL for verification. In addition to the generic TA, UPPAAL uses the concepts of *broadcast* channels for synchronizing more than two automata. The concepts of urgent and committed states are also introduced to force a transition without time delay. Similar to other model checking tools, UPPAAL can be used to verify (1) *Reachability* i.e., some condition can possibly be satisfied, (2) *Safety* i.e., some condition will never occur and (3) *Liveness* i.e., some condition will eventually become true. UPPAAL uses the concept of *templates* to provide reusability and prototyping of system components. Each template can be instantiated multiple times with varying parameters. The instantiation is called a *process*.

5.5 EAST-ADL and Timed Automata Relationship

Both timed automata (TA) and EAST-ADL are developed for real-time embedded systems. TA is a formalism which can be used for model-checking of real-time system. EAST-ADL has a broader coverage which includes but is not limited to structural and some behavioral aspects of embedded systems. There exist at least four different possibilities for relating EAST-ADL with timed automata. (1) Use TA for defining the behavior of a system by exploiting EAST-ADL external behavior representation support (Function Behavior in Fig. 5.1) as done in [12]. (2) Transform the EAST-ADL behavior description annex [16] to TA for a holistic behavioral analysis including logical, execution, nominal and error behavior. (3) Model timing constraints with timed automata with a suitable behavior abstraction. As the internal functional behavior and hence the associated constraints are out of scope of this work, only the timing constraints and design level of abstraction is considered, corresponding to option (3), with the following assumptions and limitations

- Only the *Functional Design Architecture* (FDA) is considered. The design level has two parts, namely Functional design architecture (FDA) and *Hardware Design Architecture* (HDA). While HDA covers hardware topology, FDA is used to model software components, middleware functions, device drivers and hardware transfer-functions. Hence, FDA together with constraints such as time budgets applied on its contained functions can provide a suitable abstraction for the target analysis.
- The sum of maximum execution time of all the functions having the same periodicity and allocated to same processing units is less then or equal to their period. This ensures that functions when refined to an implementation will be schedulable. For other cases, it is recommended to perform a schedulability analysis for each set of functions allocated on a single processor. This can be done by tools like Times (a TA-based tool for schedulability analysis). The deadline of a function execution is considered equal to its period.
- EAST-ADL supports hierarchical modeling (i.e., function types and prototypes) whereas UPPAAL does not have such support. Therefore, only one function type, i.e., the FDA (related to the software architecture) is allowed to have prototypes of other functions in its composition for the sake of simplicity.

5.5.1 Mapping Scheme

The following discusses a subset of mapping scheme between EAST-ADL and timed automata based on the experiences from a previous work [9]. An existing timed automata model of an emergency braking system (EBS) [8] was transformed to

EAST-ADL to derive the relationship followed by the validation of the mapping by transforming a brake-by-wire system, a representative industrial case study in EAST-ADL to timed automata. The fundamental concepts of the presented method is the main focus, therefore only a few templates, especially those which are related to the case-study in the next section are discussed. Therefore, the interested readers are referred to [10] for additional details.

The relationship is in the form of timed automata templates for function execution and timing constraints shown in Fig. 5.3. The templates for the timing constraints act as monitors indicating if a constraint is satisfied or not.

In the following figures, all the templates have their own clocks named 'Local-Clock' and all the synchronization actions and variables like MinArrivalTime, period are their input parameters in the form of bounded integer variables. The bound represents the minimum and maximum time values in the overall system specifications. The following text covers the fundamental concepts of the presented method, however, only a few templates, especially those which are related to the case-study in the next section are discussed. Therefore, the interested readers are referred to [10] for additional details.

Event: An EAST-ADL event is modeled as a synchronization action. For example, the synchronization action *output!* for the transition from *Execute* to *Init* state in Fig. 5.4b can be considered as an event corresponding to an *EventFunctionFlowPort* referring to a port with direction out or *EventFunctionClientServerPort* with kind of either *sentRequest* or *sentResponse*. The synchronization actions correspond to a simultaneous read or write on a function's ports.

Function execution behavior: As shown in Fig. 5.4, a function can be modeled with three or two locations for time-triggered and event triggered systems respectively depending on the triggering policy of the function trigger (Fig. 5.1) specified for the function under consideration. In Fig. 5.4 , the transition from *Init* to *Execute* and *Execute* to *Finished* represent the reading and writing of data on a function port respectively. The state *Execute* abstracts the logical behavior and periodicity is modeled by the *Finished* state together with its invariant and the guard of the following transition. The parameters *maxexecTime* and *minexecTime* are obtained from *ExecutionTimeConstraint* (Fig. 5.3) where *max*- and *minexecTime* correspond to the upper and lower limits of the execution time budgets. On the other hand, the period is obtained from *PeriodicEventConstraint* applied on the *EventFunction* referring to the *DesignFunctionType* under consideration.

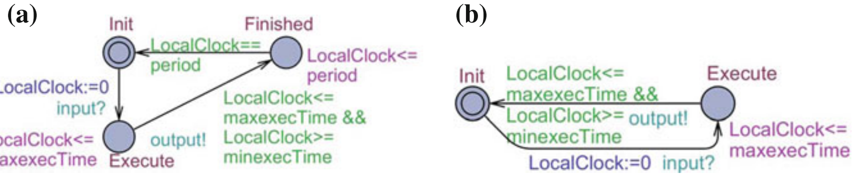

Fig. 5.4 Function templates **a** Periodic **b** Aperiodic

Timing constraints: A constraint is either satisfied or not; therefore, a minimum of four locations corresponding to initial, intermediate, success, fail states are necessary to model a constraint. On occurrence of an event, the automaton proceeds to an intermediate state. Based on the applicable guard conditions (obtained from a constraint attribute) the *fail* or *success* state is reached. The transitions to reach a *fail* or a *safe* state are enabled by clock guards and synchronization actions representing the timing bounds and event occurrences respectively. The following are two examples of the constraints applied on event and event chain respectively.

Periodic event constraint: A periodic event constraint is used to specify constraints on the periodicity of an event. An UPPAAL template for a periodic event constraint is shown in Fig. 5.5a. The three applied parameters (also shown in Fig. 5.3) are *period* (P), *jitter* (J) and the minimum arrival times of the event representing ideal, maximal and minimal time interval between occurrence of two events. The synchronization action "event?" refers to the EAST-ADL event under consideration.

Reaction constraint: A reaction constraint specifies a bound between the occurrences of stimuli and responses of an event chain. According to the EAST-ADL specifications, there exist five possible specification combinations ({upper, lower}, {upper, lower, jitter}, {upper}, {lower}, {nominal, jitter}) for the reaction constraint. The presented work considers only one combination i.e., {upper} which corresponds to the maximum time allowed between the stimulus and the response. In the reaction constraint template (Fig. 5.5b) the clock is reset when a stimulus event occurs. As soon as the response event occurs the automata transits to Fail or Success state depending on the elapsed time i.e., the 'LocalClock' value.

Rate Transition Templates: An EAST-ADL system is inherently deadlock free (Sect. 5.3.2) however, a deadlock due to blocked transitions is highly likely with the templates described earlier with functions having different execution rates. To avoid such condition, rate transition templates shown in Fig. 5.6 are introduced. The templates and the name are inspired by the Simulink[1] rate-transition block.

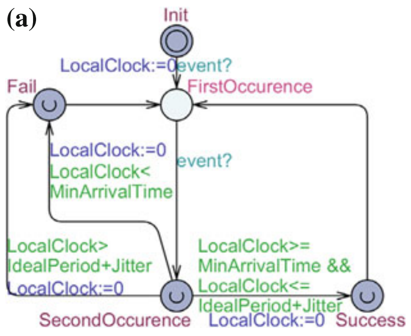

Fig. 5.5 Periodic event and reaction constraint templates

[1] http://www.mathworks.se/help/toolbox/simulink/slref/ratetransition.html

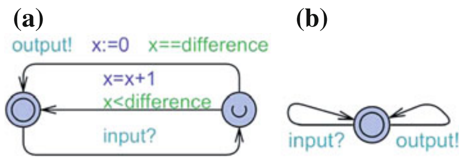

Fig. 5.6 Rate transition templates **a** Fast to slow rate transition **b** Slow to fast rate transition

The template in Fig. 5.6a is used when the sender is running at a faster rate (less period) than the receiver. The actions *input?* and *output!* correspond to the input from the sender and output to the receiver respectively. The *difference* parameter is the *difference of frequencies* obtained by dividing the period of the receiver with that of sender. For the case where the sender has low frequency, the template in Fig. 5.6b is used.

5.5.2 Usage and Automation Considerations

The following is required for the transformation and its automation:

1. A pre-defined set of templates in UPPAAL with different combination of channel types (i.e., 'chan' and 'broadcast chan' in UPPAAL) and inclusion/exclusion of synchronization actions from function temples. In this way the automation process will simply require template instantiations as UPPAAL process corresponding to EAST-ADL functions. For example, for a function in EAST-ADL representing a sensor having no input, an UPPAAL function template without 'input?' synchronization will be used. Similarly, if a constraint is applied on an EAST-ADL event or if it is transmitted to multiple receivers then the corresponding channel type in UPPAAL should be 'broadcast' type.
2. An appropriate rate transition template should be used for two functions executing at different rates and communicating with each other. This in turn requires declaration of additional channels. For example, consider a process 'A' in UPPAAL is communicating with process 'B' using channel 'channel1'. If a rate transition 'RT1' is used then an additional channel, i.e., 'channel2' is defined. In this case 'channel1' is used between 'A' and 'RT1' whereas 'channel2' is used between 'RT1' and 'B'. If the two communicating functions have the same frequency then no rate-transition process is required.
3. The minimum resolution of time has to be decided beforehand where time values are declared as a multiple of minimum time unit. For example, if the minimum unit is defined as 1 μs then 1 ms will be written as 1,000.

5.5.3 System Verification

With the introduced relationship between EAST-ADL and timed automata, the analysis becomes verification of a safety property in the following form: (1) A given constraint is satisfied *iff* for all initial conditions, the state "Fail" is never reached for all cases. (2) A system is free of any inconsistencies *iff* there is no deadlock and all the constraints are satisfied. ToS verify a given set of timing constraints of a system specification, the following two query language syntaxes have to be used.

- *A [] (not deadlock)* to verify if there exist any deadlock. Three possible reasons for a deadlock can be (1) the absence of one or more rate-transition templates in case of different frequency between two communicating functions and (2) an incorrect synchronization channel type and (3) Incompleteness of the specifications being verified.
- *A [] (not XX.Fail)* to verify that a timing constraint modeled with an UPPAAL process named XX never reaches the failed state. In case a fail state is reached, the timing constraints have one or more inconsistencies.

5.6 Brake-by-Wire Case Study

The brake-by-wire (BBW) system is a representative industrial case study. This case has been used in several EAST-ADL related projects funded by the European Commission. It provides coverage of EAST-ADL artifacts and methodology at multiple abstraction levels. A simplified version of the case study is shown in Fig. 5.7 which is a snapshot from its EAST-ADL (UML profile) implementation. For simplicity, three actuators, ABS functions and their connections are not shown in the figure.

In the simplified system, the Brake Torque Calculator (BTC) periodically (period = 10 ms) calculates the desired torque utilizing the percent value of the brake pedal. This

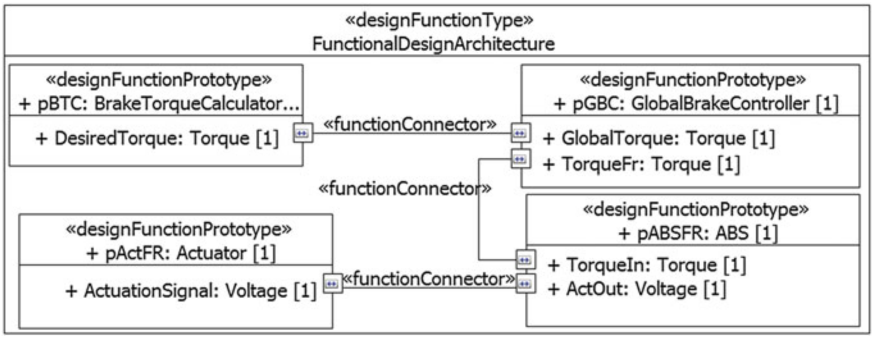
«designFunctionType»
FunctionalDesignArchitecture
«designFunctionPrototype»
+ pBTC: BrakeTorqueCalculator...
+ DesiredTorque: Torque [1]
«functionConnector»
«designFunctionPrototype»
+ pGBC: GlobalBrakeController [1]
+ GlobalTorque: Torque [1]
+ TorqueFr: Torque [1]
«functionConnector»
«designFunctionPrototype»
+ pABSFR: ABS [1]
+ TorqueIn: Torque [1]
+ ActOut: Voltage [1]
«designFunctionPrototype»
+ pActFR: Actuator [1]
+ ActuationSignal: Voltage [1]
«functionConnector»

Fig. 5.7 UML implementation of the brake-by-wire system

desired torque is utilized by the GlobalBrakeController (GBC) which periodically (period = 20 ms) calculates the torque for each wheel. The ABS functions on each wheel are responsible to control the locking of the wheel and provide the required braking torque. Both ABS and the actuators are triggered on the events related to the arrival of a data on their ports. This implies that after transformation to UPPAAL BTC and GBC will be represented by the template shown in Fig. 5.4a and the ABS by the one in Fig. 5.4b. In addition a rate transition template will also be required between BTC and GBC.

Four different events listed in Fig. 5.8 are defined for the EAST-ADL model. Furthermore, only one event chain (see also Fig. 5.2) is defined with the events 'CalculatedTorque' and 'actuation1' as the stimulus and response respectively. The constraints are listed in Fig. 5.9.

An UPPAAL model of a subset of the constraints and functions described above is shown in Fig. 5.10 where the red color represents the active locations during a simulation . The model is manually generated. Compare the process names, synchronization actions and constraints, i.e., PCC1, RC1 and PEC1 with the discussion and figures earlier in this section.

Fig. 5.10 shows three constraints i.e., PEC1 (Periodic Event Constraint), PCC1 (Precedence Constraint) and RC1 (Reaction Constraint). To verify the consistency the CTL syntax as described in Section 5.5.3 were used. For example, *A[] (not deadlock)* to check if the system is deadlock free and *A[] (not PEC1.Fail)* to check if

Event Name	Type	Attributes
BTCTriggerEvent	EventFunction	TargetFunctionPrototype=pBTC
GBCTriggerEvent	EventFunction	TargetFunctionPrototype=pGBC

CalculatedTorque	EventFunctionFlowPort	TargetFunctionFlowPort=DesiredTorque
		TargetFunctionPrototype=pBTC

actuation1	EventFunctionFlowPort	TargetFunctionFlowPort=ActOut
		TargetFunctionPrototype=pABSFR

Fig. 5.8 Events

Constraint Name	Type	Attributes
BTCExecution	Execution time	TargetFunctionPrototype=pBTC , Lower = 3 , Upper =5
GBCExecution	Execution time	TargetFunctionPrototype=pGBC, Lower = 2, Upper =6
ABSFRExecution	Execution time	TargetFunctionPrototype=pABSFR, Lower = 2, Upper =3
ABSFLExecution	Execution time	TargetFunctionPrototype=pABSFL, Lower = 2, Upper =3
ABSRRExecution	Execution time	TargetFunctionPrototype=pABSRR, Lower = 2, Upper =3
ABSRLExecution	Execution time	TargetFunctionPrototype=pABSRL, Lower = 2, Upper =3
RC1	Reaction	Scope = EC1, ReactionTime = 50 ms
PEC1	Periodic event	TargetEvent= CaclulatedTorque, MinArrivalTime= 3ms, IdealPeriod = 10 ms, Jitter = 9 ms
PCC1	Precedence	Preceding = CalculatedTorque, Successive = actuation1

Fig. 5.9 Timing and event constraints

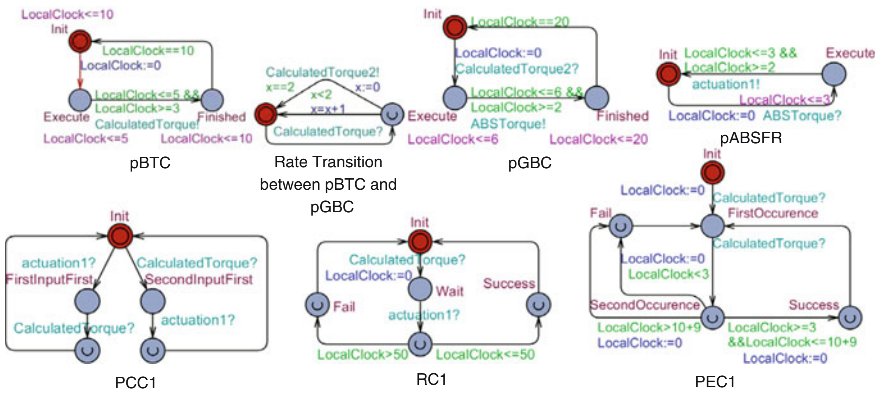

Fig. 5.10 Brake-by-wire model in UPPAAL

the periodic event constraint is satisfied. All the properties were found to be satisfied using the UPPAAL verifier. The time taken for the verification was less than a second.

5.7 Discussion

A method to analyze consistency of timing constraints specified using EAST-ADL is presented. The proposed mapping scheme is a basis for automated transformations between EAST-ADL and timed automata based tools. We have experimented with different model transformation technologies such as Atlas Transformation Language (ATL) or Model Query Language (MQL) [9]. Transformation of EAST-ADL specifications is a complex and non-trivial task due to the distribution of information among its extensions. One of the main benefits of the presented approach is that it reduces the effort related to model-transformation and related complexity. Instead of creating a new TA process for every timing constraint, the transformation requires only instantiation of the templates. This is especially true for imperative transformation languages like MERL (MetaEdit+ Reporting Language). Furthermore, the template-based approach is also a scalable solution where the templates can be extended for as many events as required to be considered for an event chain. The extension mechanisms are discussed in [10].

While there exist otherapproaches targeting different types of constraints for example, dependency graph for checking function precedence [17], the presented work is suitable in the context of MBD. The use of TA can enable bridging the gap between the design and testing phase. Generation of test cases from the timed automata analysis of EAST-ADL specifications for carrying out testing at different design phases is one of the future possible extensions.

At first glance, the use of techniques like scheduling analysis might seem to be more appropriate than the use of timed automata. The use of such techniques are applicable at the implementation level of EAST-ADL which in turn corresponds to the AUTOSAR standard [18] instead of the design level of abstraction addressed by this chapter. To enable analysis such as schedulability, the design level of abstraction has be augmented with additional details as presented in [19]. In addition, the concurrency of functions at the design level of abstraction and the fact that there exists an *n-to-m* relationship between EAST-ADL functions and AUTOSAR software components as well as runnables [20, 21] further motivates the use of timed automata based analysis.

In Sect. 5.5, three different possibilities for relating EAST-ADL and timed automata were mentioned. Based on the experiences from the work presented in this chapter, the possibility of transforming the EAST-ADL behavior description annex (BDA) to timed automata has also been studied [14]. The BDA provides support for relating different kind of behavior such as execution behavior and timing constraints considered in this chapter, logical behavior, and error behavior specified by the EAST-ADL's dependability modeling extension. The templates described in this chapter can serve as a monitor to verify different types of timed behavioral.

The main focus of the work presented in this chapter was on exploring a relationship between EAST-ADL and timed automata as a step towards an integrated environment providing architecture-centric development of automotive embedded systems in a seamless manner. This also implies that aspects like state space coverage were not the major focus however, discussed in [14]. The timed automata semantics and the assumptions presented in this chapter pose some limitations on EAST-ADL modeling and transformation work. For example, the function templates assumes one input and one output port which in turn requires that a function can only be triggered by arrival of data on one port only etc. in contrast to EAST-ADL support for multiple triggers. Another issue related to the fast to slow rate transition template is that it is only applicable for the cases where the frequencies are multiples of each other. Similarly, aspects like jitter are considered to a limited extent. Efforts to overcome this limitations are one of the planned future activities.

Other issueswhich require further investigation are automatic test-case generation directly from EAST-ADL specifications, combined analysis of internal and execution behavior and methods for transferring the analysis results back for storing them as part of the EAST-ADL model using its verification and validation extension.

References

1. Törngren, M., Chen, D., Malvius, D., Axelsson, J.: Automotive embedded systems handbook. In: Model-Based Development of Automotive Embedded Systems. CRC Press, Boca Raton, FL (2009)
2. Lönn, H., Freund, U.: Automotive embedded systems handbook. In: Automotive Architecture Description Languages, CRC Press, Boca Raton, FL (2009)
3. Broy, M., Feilkas, M., Herrmannsdoerfer, M., Merenda, S., Ratiu, D.: Seamless model-based development: from isolated tools to integrated model engineering environments. Proc. IEEE **98**(4), 526–545 (2010)

4. The ATESST2 Consortium: EAST ADL 2.0 Specification. Project Deliverable D4.1.1. http://www.east-adl.info/repository/EAST-ADL2.1/EAST-ADL-Specification_2010-06-30.pdf (2010)
5. Bengtsson, J., Yi, W.: Timed automata: semantics, algorithms and tools. In: Lectures on Concurrency and Petri Nets, LNCS, vol. 3098, pp. 87–124. Springer, Berlin (2004)
6. Hendriks, M., Verhoef, M.: Timed automata based analysis of embedded system architectures. In: Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th, International, p. 8 (2006)
7. Ponsard, C., Massonet, P., Molderez, J.F., Rifaut, A., Lamsweerde, A.V., Van, H.T.: Early verification and validation of mission critical systems. Form. Methods Syst. Des. **30**(3), 233–247 (2007)
8. Montag, P., Nowotka, D., Levi, P.: Verification in the design process of large real-time systems: a case study. In: Automotive Safety and Security 2006, Stuttgart (Germany), October 12–13, 2006, pp. 1–13 (2006)
9. Qureshi, T.N., Chen, D.J., Persson, M., Törngren, M.: Towards the integration of UPPAAL for formal verification of EAST-ADL timing constraint specification. In: Workshop on Time Analysis and Model-Based Design, from Functional Models to Distributed Deployments (2011)
10. Qureshi, T.N., Chen, D.J., Törngren, M.: A timed automata-based method to analyze EAST-ADL timing constraint specifications. In: Modelling Foundations and Applications, Lecture Notes in Computer Science, vol. 7349, pp. 303–318. Springer, Berlin (2012)
11. Feng, L., Chen, D.J., Lönn, H., Törngren, M.: Verifying system behaviors in EAST-ADL2 with the SPIN model checker. In: Mechatronics and Automation (ICMA), 2010 International Conference on, pp. 144–149 (2010)
12. Kang, E.Y., Schobbens, P.Y., Pettersson, P.: Verifying functional behaviors of automotive products in EAST-ADL2 Using UPPAAL-PORT. In: Computer Safety, Reliability, and Security, LNCS, vol. 6894, pp. 243–256. Springer, Berlin (2011)
13. Enoiu, E.P., Marinescu, R., Seceleanu, C., Pettersson, P.: ViTAL : A verification tool for EAST-ADL models using UPPAAL PORT. In: Proceedings of the 17th IEEE International Conference on Engineering of Complex Computer Systems. IEEE Computer Society Press (2012)
14. Qureshi, T.N.: Enhancing model-based development of embedded systems: modeling, simulation and model-transformation in an automotive context. Trita-mmk, issn 1400–1179; 2012:16, isbn 978-91-7501-465-4, Department of Machine Design, KTH - The Royal Institute of Technology, Sweden (2012)
15. Stappert, F., Jonsson, J., Mottok, J., Johansson, R.: A design framework for End-To-End timing constrained automotive applications. Embedded Real-Time Software and Systems (ERTS'10) (2010)
16. The MAENAD consortium: language concepts supporting engineering scenarios. Project Deliverable D3.1.1. http://www.maenad.eu/publications.html (2012)
17. Zhao, Y.: On the design of concurrent, distributed real-time systems. Ph.D. thesis, EECS Department, University of California, Berkeley (2009)
18. AUTOSAR Consortium: AUTomotive Open System ARchitecture. http://www.autosar.org
19. Anssi, S., Tucci-Pergiovanni, S., Mraidha, C., Albinet, A., Terrier, F., Gerard, S.: Completing EAST-ADL2 with MARTE for Enabling Scheduling Analysis for Automotive Applications. Conference ERTS, In (2010)
20. Cuenot, P., Frey, P., Johansson, R., Lönn, H., Reiser, M.O., Servat, D., Tavakoli Kolagari, R., Chen, D.: Developing automotive products using the EAST-ADL2, an AUTOSAR compliant architecture description language. In: Proceedings of the 4th European Congress ERTS (Embedded Real Time Software) (2008)
21. Qureshi, T.N., Chen, D., Lönn, H., Törngren, M.: From EAST-ADL to AUTOSAR. Tech. Rep. TRITA-MMK 2011:12, ISSN 1400–1179, ISRN/KTH/MMK/R-11/12-SE, Mechatronics Lab, Department of Machine Design, KTH, Stockholm, Sweden (2011)